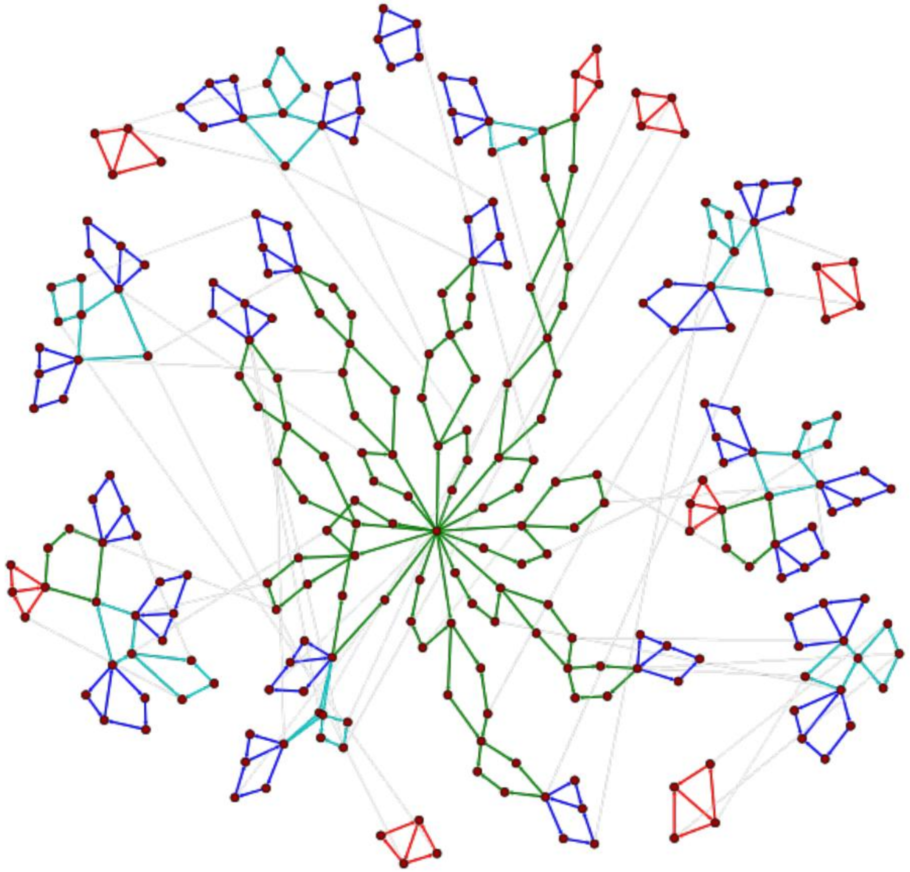
Motif, Isomorphism, Graphlets

CE642: Social and Economic Networks
Maryam Ramezani
Sharif University of Technology
maryam.ramezani@sharif.edu

01

Motif

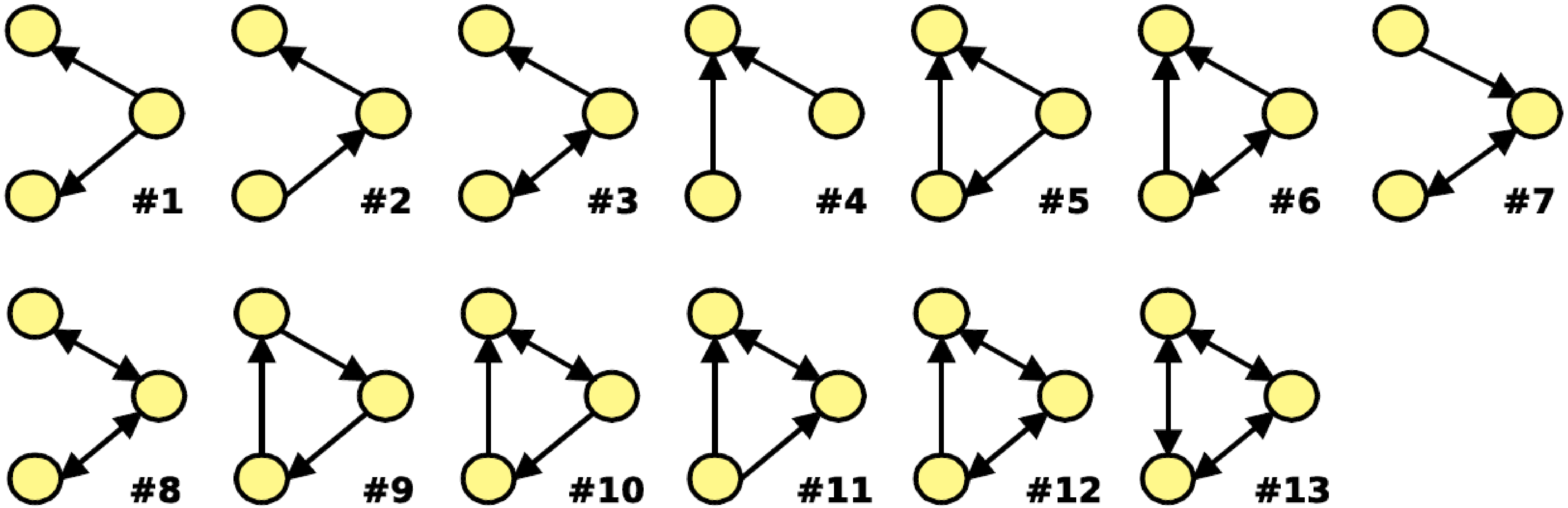
Building Blocks of Networks



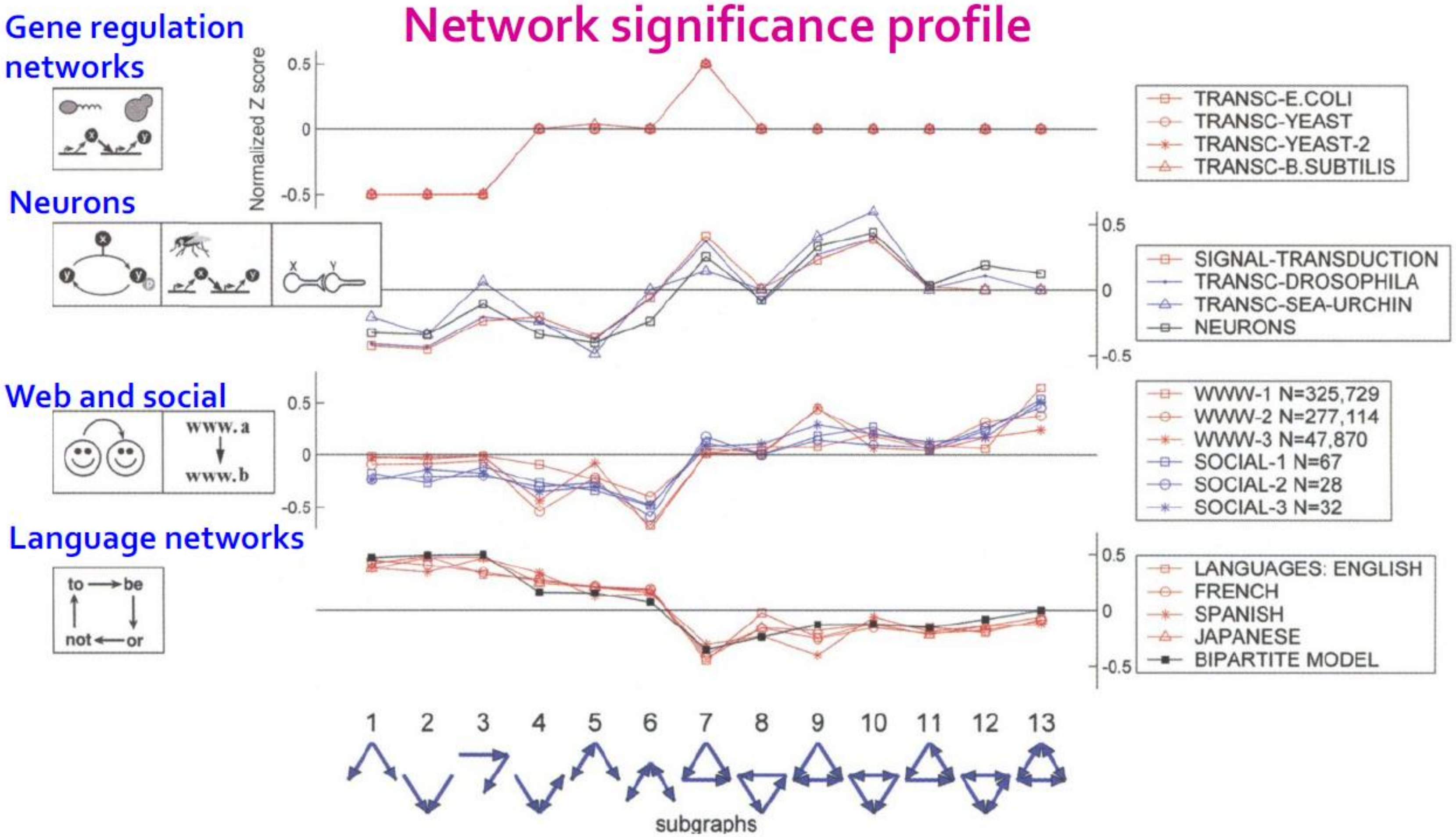
Subgraph decomposition of an electronic circuit

Case Example of Subgraphs

Let’s consider all possible (non-isomorphoic) directed subgraphs of size 3



Case Example of Subgraphs



Networks from the same domain have similar significance profiles

Network Motifs

- **Network motifs:** "recurring, significant patterns of interconnections"

- **How to define a network motif:**
  - **Pattern:** Small induced subgraph
  - **Recurring:** Found many times, i.e., with high frequency
  - **Significant:** More frequent than expected, i.e., in randomly generated networks
    - Erdos-Renyi random graphs, scale-free networks

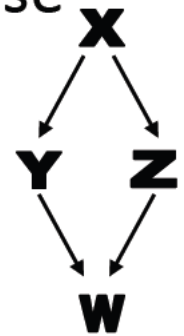
Why Do We Need Motifs?

- **Motifs:**
  - Help us understand how networks work
  - Help us predict operation and reaction of the network in a given situation

Feed-forward loop

- **Examples:**
  - **Feed-forward loops:** found in networks of neurons, where they neutralize “biological noise”
  - **Parallel loops:** found in food webs
  - **Single-input modules:** found in gene control networks

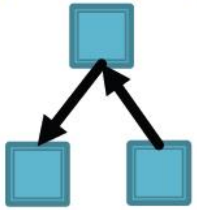
Single-input module

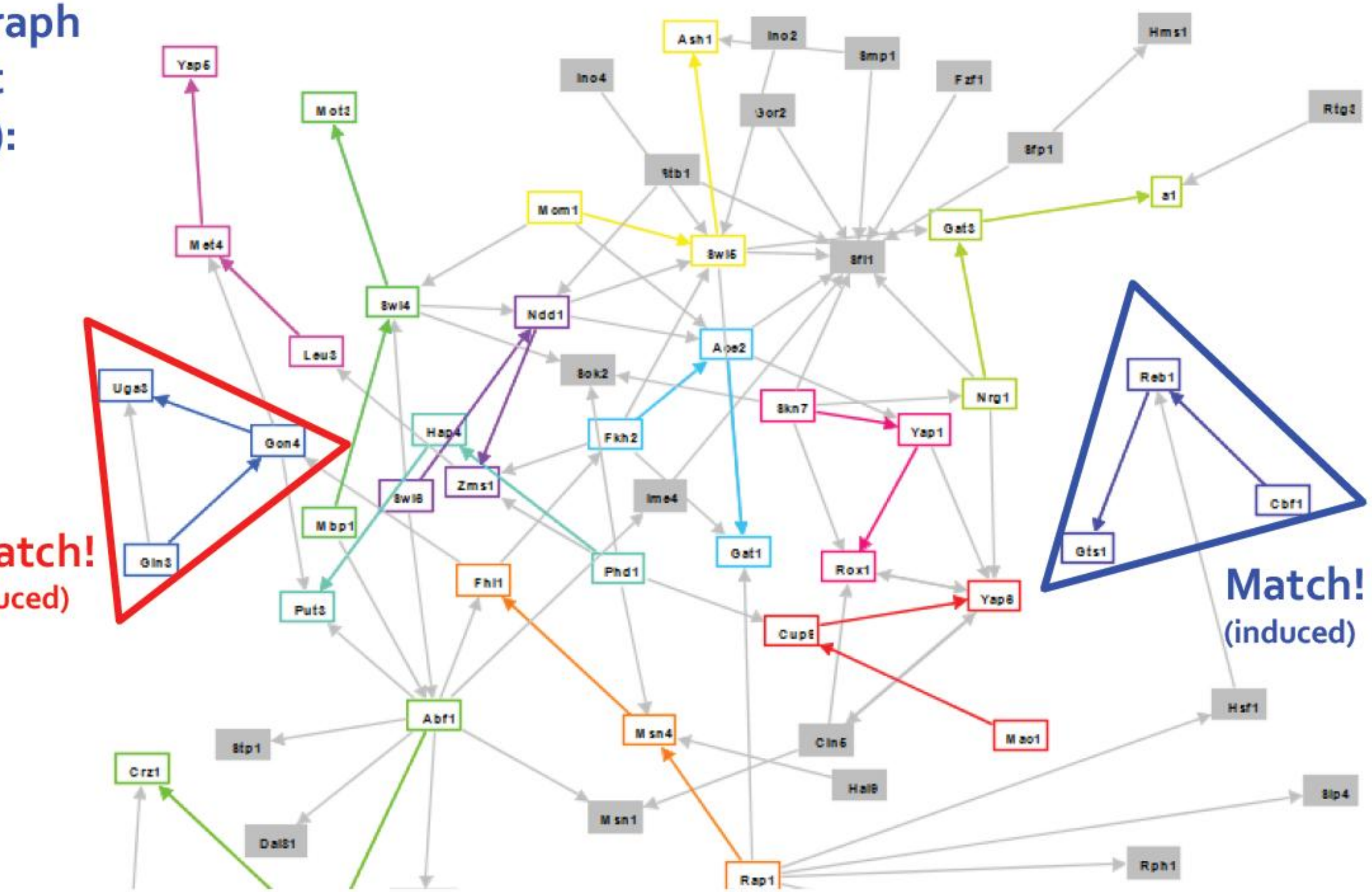


Parallel loop

Motifs: Induced Subgraphs

Induced subgraph of interest (aka Motif):

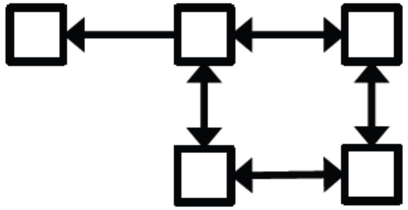




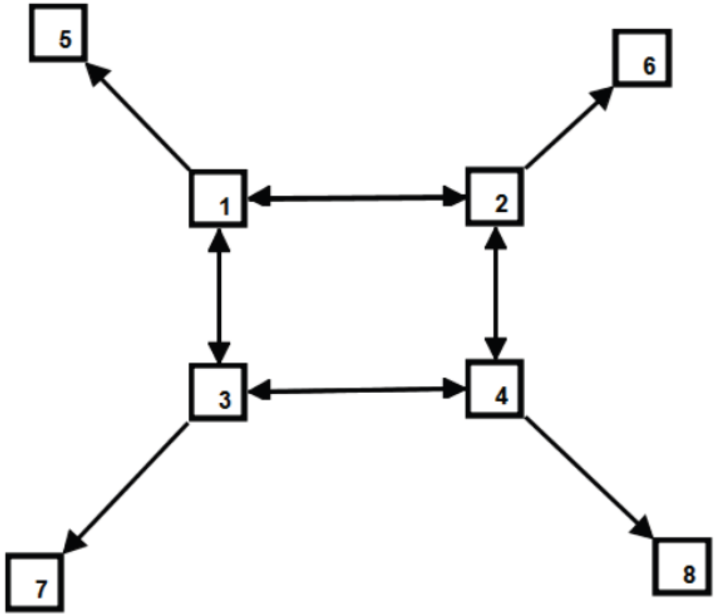
Induced subgraph of graph G is a graph, formed from a subset X of the vertices of graph G and all of the edges connecting pairs of vertices in subset X.

Motifs: Recurrence

Motif of interest:



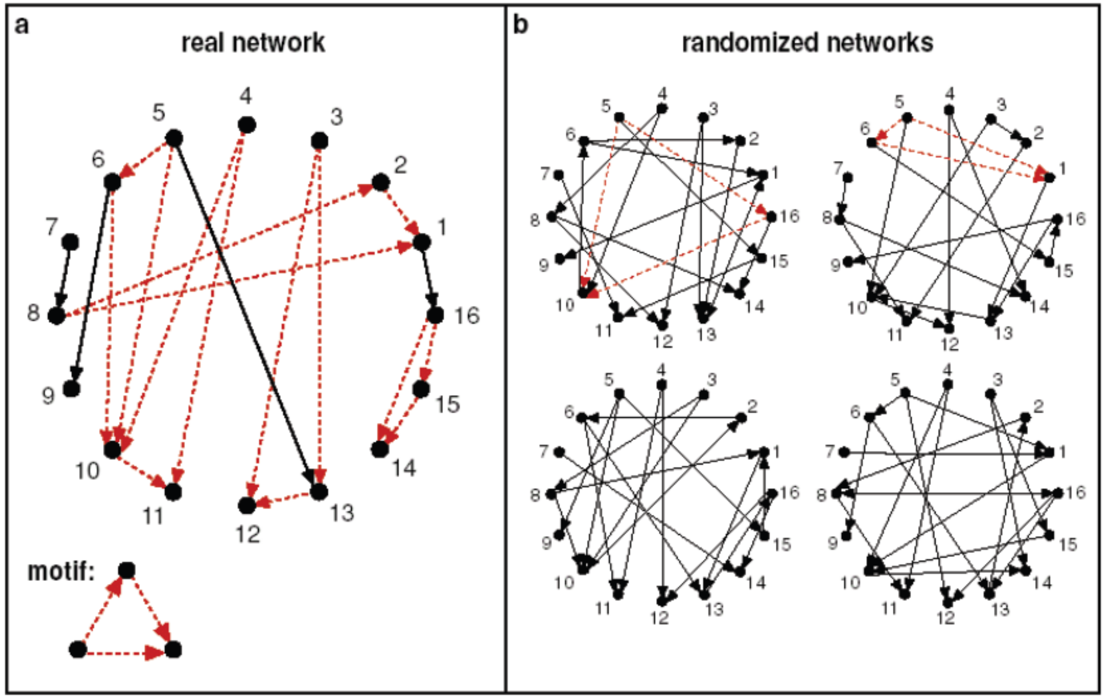
- Allow **overlapping of motifs**
- Network on the right has 4 occurrences of the motif:
  - {1,2,3,4,5}
  - {1,2,3,4,6}
  - {1,2,3,4,7}
  - {1,2,3,4,8}



Example borrowed from Pedro Ribeiro

Significance of a Motif

- **Key idea:** Subgraphs that occur in a real network much more often than in a random network have functional significance



Significance of a Motif

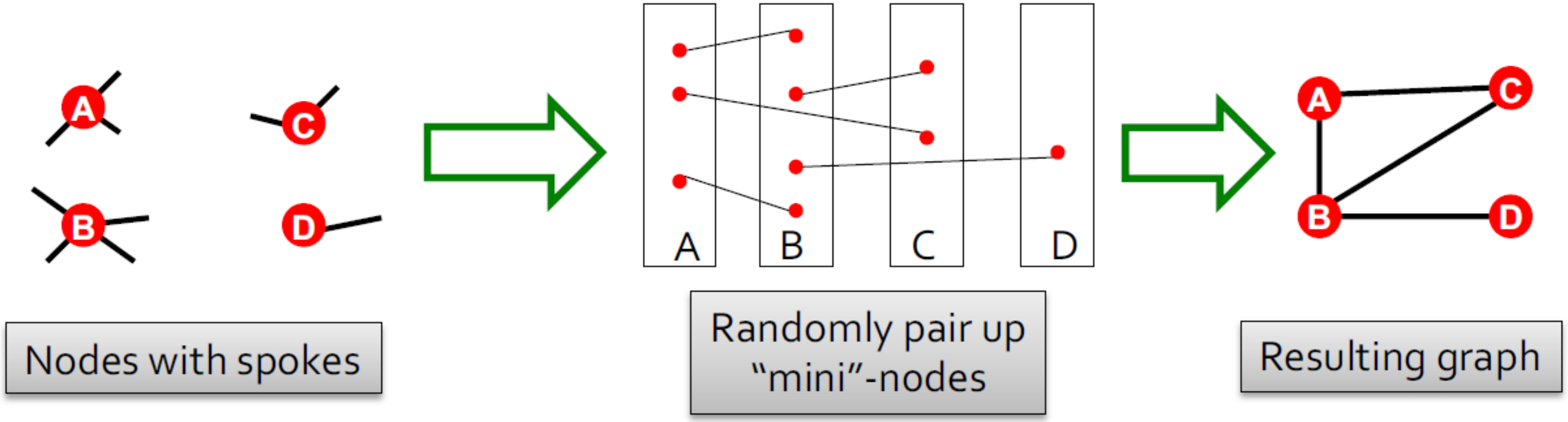
- Motifs are **overrepresented** in a network when compared to **randomized networks:**
  - Z_i captures **statistical significance of motif i**:
    $Z_i = (N_i^{\text{real}} - \bar{N}_i^{\text{rand}})/\text{std}(N_i^{\text{rand}})$
    - N_i^{real} is #(subgraphs of type i) in network G^{real}
    - N_i^{rand} is #(subgraphs of type i) in randomized network G^{rand}
- **Network significance profile (SP):**

$$SP_i = Z_i / \sqrt{\sum_j Z_j^2}$$

  - SP is a vector of **normalized Z-scores**
  - SP emphasizes relative significance of subgraphs:
    - Important for comparison of networks of different sizes
    - Generally, larger networks display higher Z-scores

Configuration Model

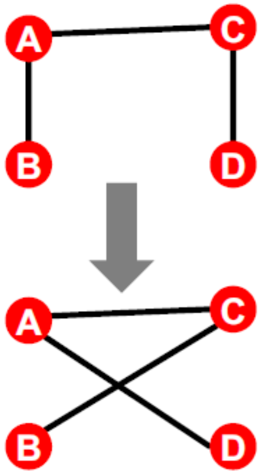
- **Goal:** Generate a random graph with a given degree sequence $k_1, k_2, \ldots k_N$
- **Useful as a "null" model of networks:**
  - We can compare the real network G^{real} and a "random" G^{rand} which has the same degree sequence as G^{real}
- **Configuration model:**



We ignore double edges and self-loops when creating the final graph

Construct Random Graph

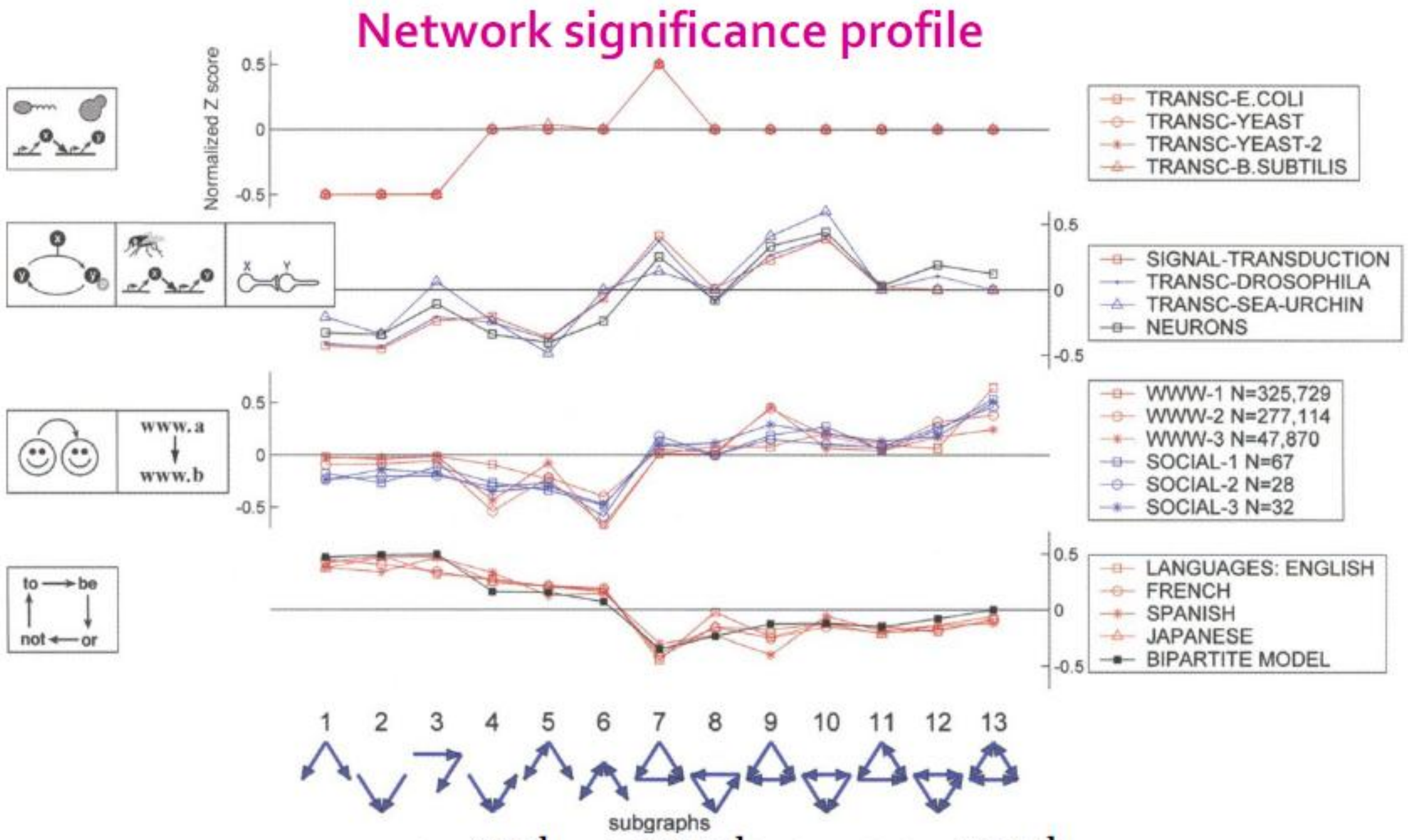
- Start from a **given graph** G
- Repeat **the switching step** $Q \cdot |E|$ times:
  - Select a pair of edges A→B, C→D at random
  - **Exchange** the endpoints to give A→D, C→B
    - Exchange edges only if no multiple edges or self-edges are generated



- **Result:** A randomly rewired graph:
  - Same node degrees, randomly rewired edges

- Q is chosen large enough *(e.g.,* $Q = 100$*)* for the process to converge

Motifs: Significance Example

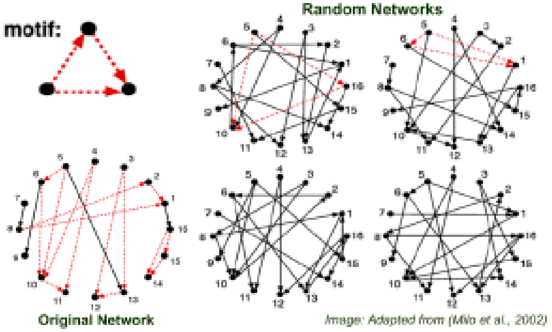


$$Z_i = (N_i^{\text{real}} - \bar{N}_i^{\text{rand}})/\text{std}(N_i^{\text{rand}})$$

Detecting Motifs

- Count subgraphs i in G^{real}
- Count subgraphs i in random networks G^{rand}:
  - **Configuration model:** Each G^{rand} has the same #(nodes), #(edges) and #(degree distribution) as G^{real}

- Assign **Z-score** to i:
  - $Z_i = (N_i^{\text{real}} - \bar{N}_i^{\text{rand}}) / \text{std}(N_i^{\text{rand}})$
  - **High Z-score:** Subgraph i is a **network motif of G**

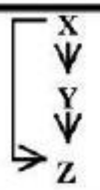
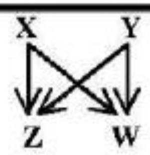
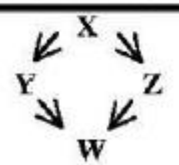
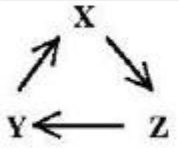
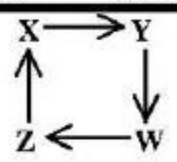
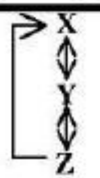
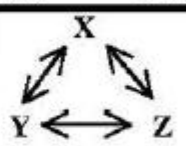
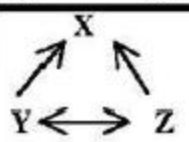


Motif Examples

Network	Nodes	Edges	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score
Gene regulation (transcription)					**Feed-forward loop**			**Bi-fan**			
E. coli	424	519	40	7 ± 3	10	203	47 ± 12	13			
*S. cerevisiae**	685	1,052	70	11 ± 4	14	1812	300 ± 40	41			
Neurons					**Feed-forward loop**			**Bi-fan**			**Bi-parallel**
C. elegans†	252	509	125	90 ± 10	3.7	127	55 ± 13	5.3	227	35 ± 10	20
Food webs					**Three chain**			**Bi-parallel**			
Little Rock	92	984	3219	3120 ± 50	2.1	7295	2220 ± 210	25			
Ythan	83	391	1182	1020 ± 20	7.2	1357	230 ± 50	23			
St. Martin	42	205	469	450 ± 10	NS	382	130 ± 20	12			
Chesapeake	31	67	80	82 ± 4	NS	26	5 ± 2	8			
Coachella	29	243	279	235 ± 12	3.6	181	80 ± 20	5			
Skipwith	25	189	184	150 ± 7	5.5	397	80 ± 25	13			
B. Brook	25	104	181	130 ± 7	7.4	267	30 ± 7	32			

Z-scores of individual motifs for different networks

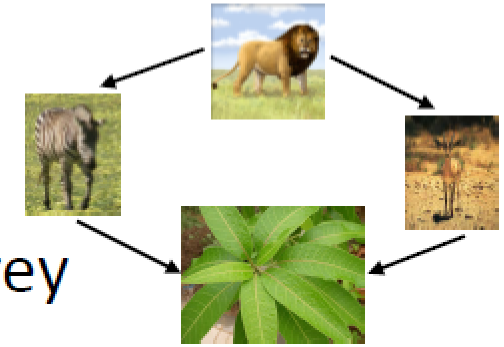
Motif Examples

Network	Nodes	Edges	N_{real}	$N_{rand} \pm$ SD	Z score	N_{real}	$N_{rand} \pm$ SD	Z score	N_{real}	$N_{rand} \pm$ SD	Z score
Electronic circuits (forward logic chips)					**Feed-forward loop**			**Bi-fan**			**Bi-parallel**
s15850	10,383	14,240	424	2 ± 2	285	1040	1 ± 1	1200	480	2 ± 1	335
s38584	20,717	34,204	413	10 ± 3	120	1739	6 ± 2	800	711	9 ± 2	320
s38417	23,843	33,661	612	3 ± 2	400	2404	1 ± 1	2550	531	2 ± 2	340
s9234	5,844	8,197	211	2 ± 1	140	754	1 ± 1	1050	209	1 ± 1	200
s13207	8,651	11,831	403	2 ± 1	225	4445	1 ± 1	4950	264	2 ± 1	200
Electronic circuits (digital fractional multipliers)					**Three-node feedback loop**			**Bi-fan**			**Four-node feedback loop**
s208	122	189	10	1 ± 1	9	4	1 ± 1	3.8	5	1 ± 1	5
s420	252	399	20	1 ± 1	18	10	1 ± 1	10	11	1 ± 1	11
s838‡	512	819	40	1 ± 1	38	22	1 ± 1	20	23	1 ± 1	25
World Wide Web					**Feedback with two mutual dyads**			**Fully connected triad**			**Uplinked mutual dyad**
nd.edu§	325,729	1.46e6	1.1e5	2e3 ± 1e2	800	6.8e6	5e4±4e2	15,000	1.2e6	1e4 ± 2e2	5000

Z-scores of individual motifs for different networks

Motif Examples

- **Network of neurons and a gene network** contain similar motifs:
  - Feed-forward loops and bi-fan structures
  - Both are information processing networks with sensory and acting components
- **Food webs** have parallel loops:
  - Prey of a particular predator share prey



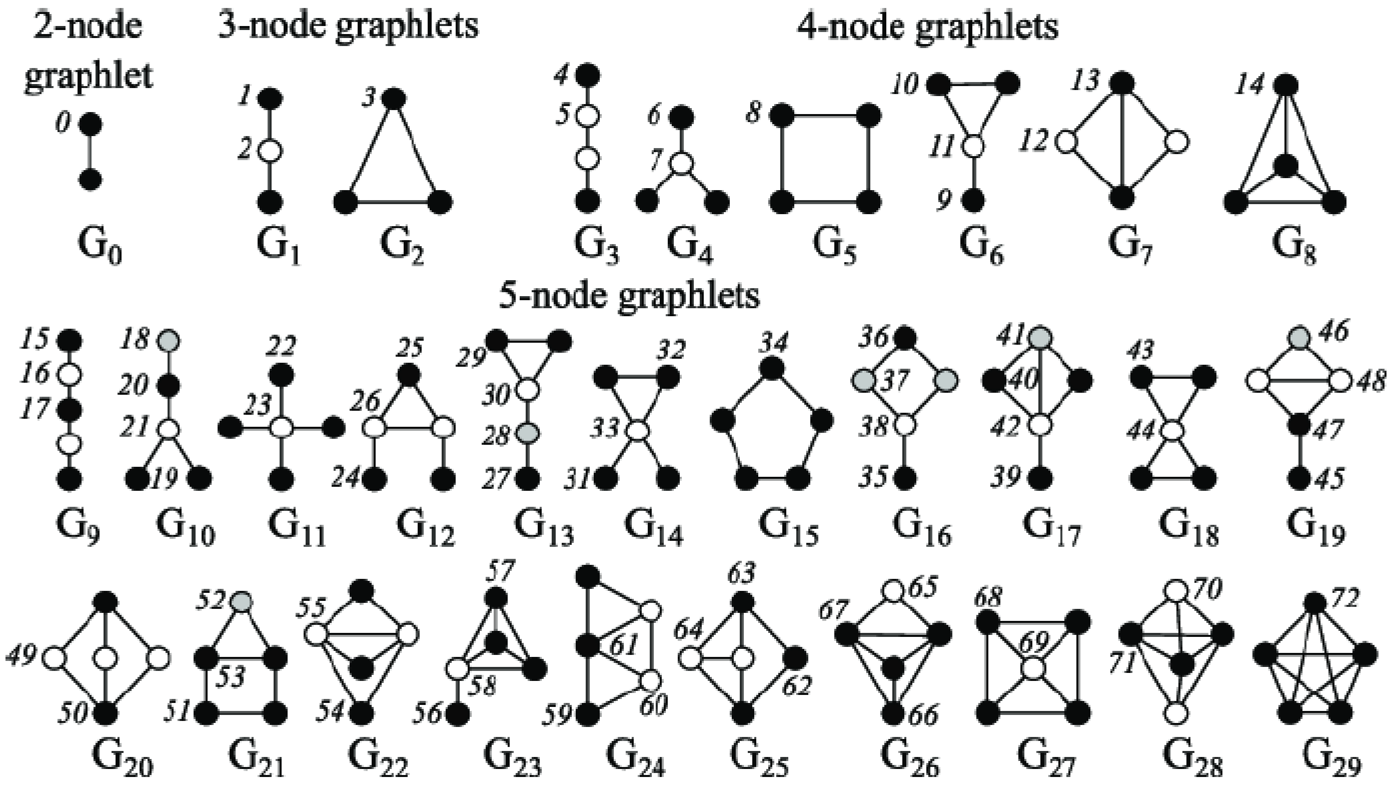
- **WWW network** has bidirectional links
  - Design that allows the shortest path between sets of related pages

02

Graphlets: Node Feature Vectors

Graphlets

- **Graphlets:** connected non-isomorphic subgraphs
  - **Induced subgraphs of any frequency**



For $n = 3, 4, 5, \ldots 10$ there are 2, 6, 21, … 11716571 graphlets!

Motif vs Graphlet

Graphlet Degree Vector

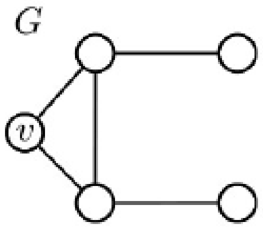
- **Next:** Use graphlets to obtain a **node-level** subgraph metric

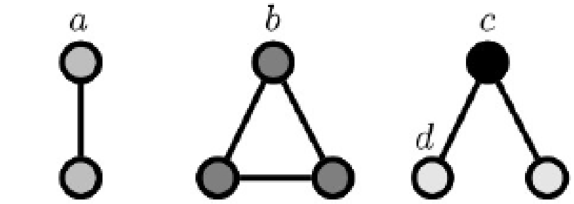
- **Degree** counts **#(edges)** that a node touches:
  - Can we generalize this notion for graphlets? – Yes!

- **Graphlet degree vector** counts **#(graphlets)** that a node touches

Automorphism Orbits

- An **automorphism orbit** takes into account the symmetries of a subgraph
- **Graphlet Degree Vector (GDV):** a vector with the frequency of the node in each **orbit position**
- **Example: Graphlet degree vector of node *v***

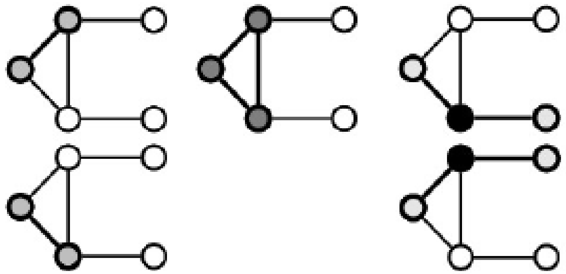




For a node u of graph G, the automorphism orbit of u is $Orb(u) = \{v \in V(G); v = f(u) \text{ for some } f = \mathrm{Aut}(G)\}$.

The Aut denotes an automorphism group of G, *i.e.*, an isomorphism from G to itself.

$orbit$	a	b	c	d
$GDV(v)$	2	1	0	2

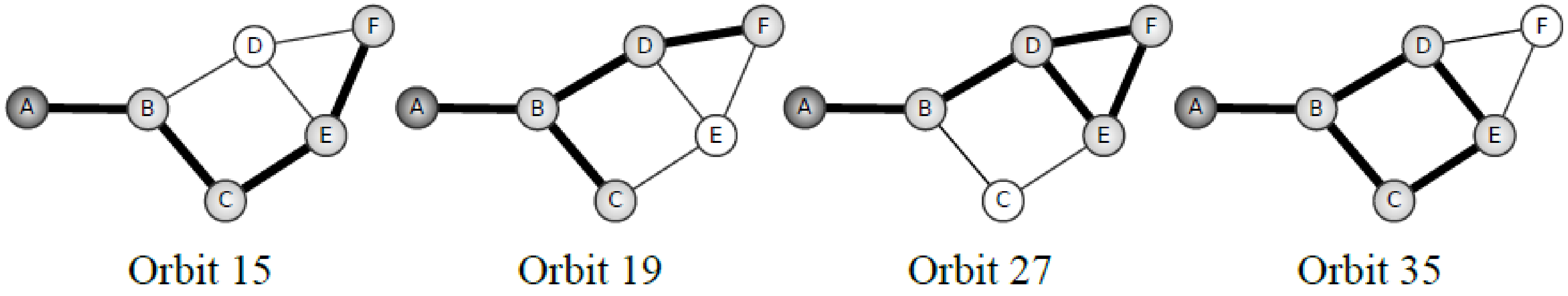


Graphlet Degree Vector (GDV)

- Graphlet degree vector **counts #(graphlets)** that a node touches **at a particular orbit**
- **Considering** graphlets on 2 to 5 nodes we get:
  - **Vector of 73 coordinates** is a signature of a node that describes the topology of node's neighborhood
  - Captures its interconnectivities out to a **distance of 4** hops

- Graphlet degree vector provides a measure of a **node's local network topology**:
  - Comparing vectors of two nodes provides a highly constraining measure of local topological similarity between them

Graphlet Degree Vector (GDV)



Orbit	0	1	2...3	4	5	6	7...14	15	16...18	19	20...26	27	28...34	35	36...72
GDV(A)	1	2	0...0	3	0	1	0...0	1	0...0	1	0...0	1	0...0	1	0...0

Graphlet Degree Vector (GDV) of node A:

- i-th element of GDV(A): #(graphlets) that touch A at orbit i
- Highlighted are graphlets that touch node A at orbits 15, 19, 27, and 35 from left to right

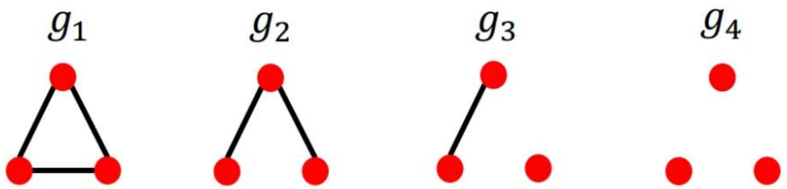
Graphlet Kernels

- The idea of graphlet kernel is to count the number of graphlets in a graph like we did in Graphlet Degree Vector. Although the idea of graphlet here is slightly different. Here, graphlets need not to be disconnected and not rooted as well.

Graphlet Kernels

- Below example shows how to count graphlet for 3 node subgraph:

For $k = 3$, there are 4 graphlets.

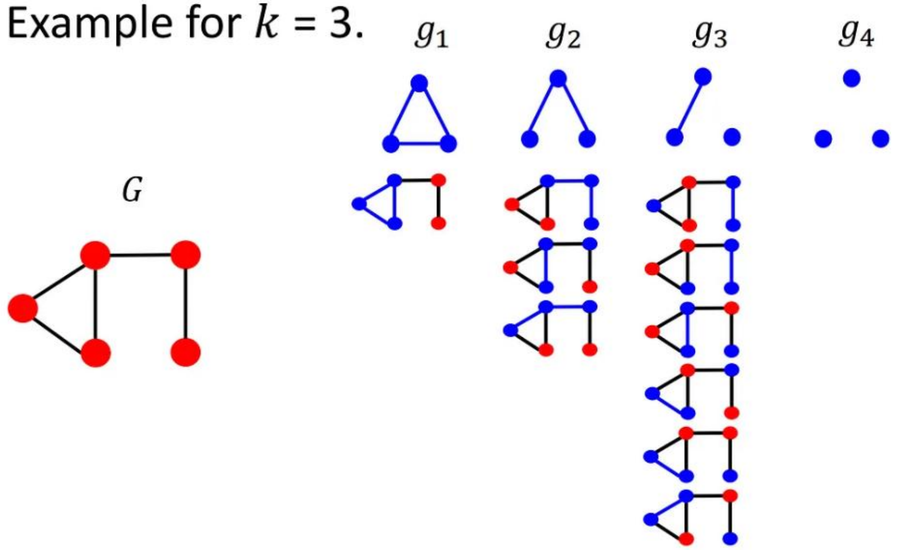


So, how to create graphlet kernel with this approach. Assume you have a graph **G**, then the graphlet vector $\boldsymbol{G_j}$ can be defined as graphlet count vector:

$$(f\boldsymbol{g})_i = \#(g_i \subseteq G) \text{ for } i = 1,2,\ldots,n_j$$

Example

- Example for $k = 3$.



$$\boldsymbol{f}_G = (1, \quad 3, \quad 6, \quad 0)^{\mathrm{T}}$$

Then, if you have two graphs *G* and *G'*, you can calculate graphlet kernel with the below formula:

$$K(G, G') = \boldsymbol{f}_G{}^{\mathrm{T}} \boldsymbol{f}_{G'}$$

03

Finding Motifs & Graphlets

Algorithms

- Finding size-k motifs/graphlets requires solving two challenges:
  - **1) Enumerating** all size-k connected subgraphs
  - **2) Counting** #(occurrences of each subgraph type)

- **Just knowing if a certain subgraph exists in a graph is a hard computational problem!**
  - Subgraph isomorphism is NP-complete

- **Computation time grows exponentially as the size of the motif/graphlet increases**
  - Feasible motif size is usually small (3 to 8)

Courting Subgraphs

- **Network-centric approaches:**
  - **1) Enumerating** all size-k connected subgraphs
  - **2) Counting** #(occurrences of each subgraph type) via graph **isomorphisms test**

Exact Subgraph Enumeration (ESU)

- **Two sets:**
  - V_{subgraph} : currently constructed subgraph (motif)
  - $V_{\text{extension}}$: set of candidate nodes to extend the motif
- **Idea:** Starting with a node v, add those nodes u to $V_{\text{extension}}$ set that have two properties:
  - u's node_id must be larger than that of v
  - u may only be neighbored to some newly added node w but not of any node already in V_{subgraph}
- ESU is implemented as a **recursive function:**
  - The running of this function can be displayed as a **tree-like structure of depth k**, called the **ESU-Tree**

Exact Subgraph Enumeration (ESU)

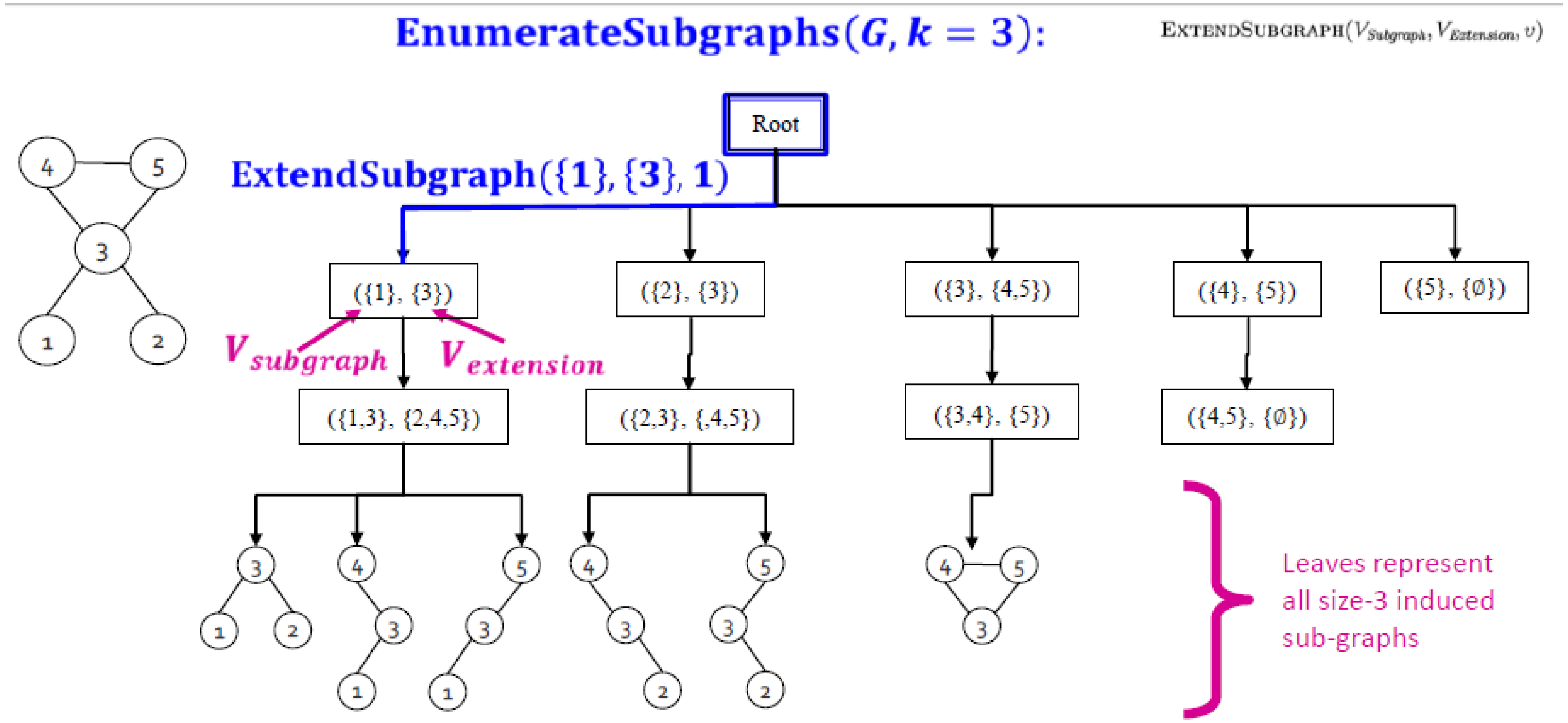
Algorithm: $\textsc{EnumerateSubgraphs}(G, k)$ (ESU)
Input: A graph $G = (V, E)$ and an integer $1 \leq k \leq |V|$.
Output: All size-k subgraphs in G.

for each vertex $v \in V$ **do**
    $V_{Extension} \leftarrow \{u \in N(\{v\}) \, : \, u > v\}$
    call $\textsc{ExtendSubgraph}(\{v\}, V_{Extension}, v)$
return

$\textsc{ExtendSubgraph}(V_{Subgraph}, V_{Extension}, v)$
if $|V_{Subgraph}| = k$ **then output** $G[V_{Subgraph}]$ and **return**
while $V_{Extension} \neq \emptyset$ **do**
    Remove an arbitrarily chosen vertex w from $V_{Extension}$
    $V'_{Extension} \leftarrow V_{Extension} \cup \{u \in N_{excl}(w, V_{Subgraph}) \, : \, u > v\}$
    call $\textsc{ExtendSubgraph}(V_{Subgraph} \cup \{w\}, V'_{Extension}, v)$
return

$N_{excl}(w, V_{Subgraph}) = N(w) \setminus (V_{Subgraph} \cup N(V_{Subgraph}))$ is exclusive neighborhood: All nodes neighboring w but not of $V_{Subgraph}$ or $N(V_{Subgraph})$

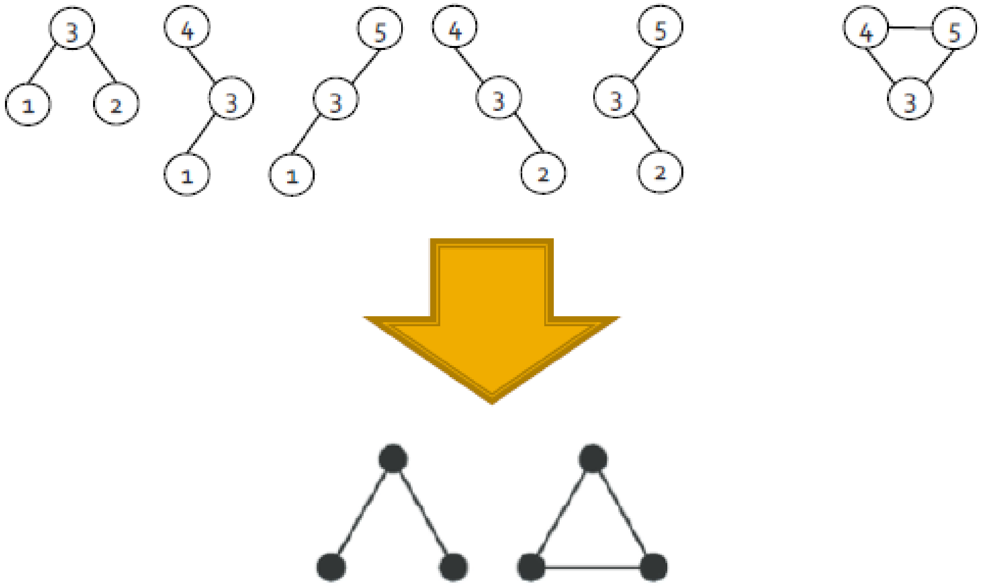
ESU-Tree Example



- Nodes in the ESU-tree include two adjoining sets:
  - $V_{subgraph}$: Current subgraph (a set of adjacent nodes)
  - $V_{extension}$: Nodes adjacent to $V_{subgraph}$ whose node_ids are larger than starting node v

Use ESU-Tree to Count Subgraphs

- **So far,** we **enumerated all size-k subgraphs** in the input graph
- **Next step: Count the graphs**



Count: 5 1

Use ESU-Tree to Count Subgraphs

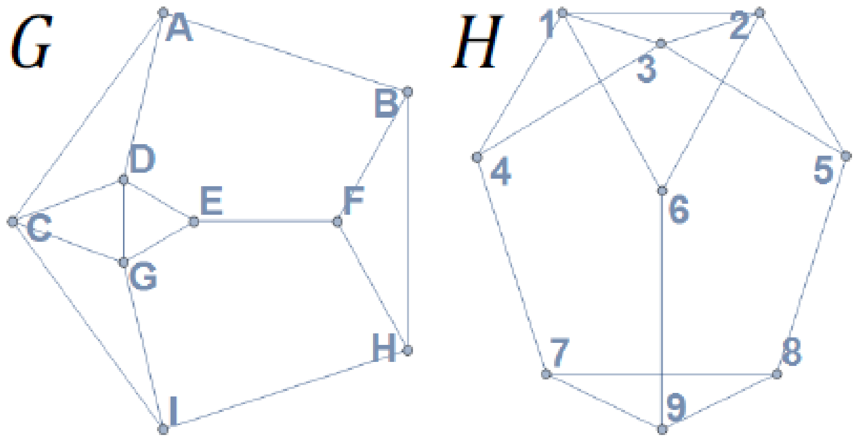
- **So far,** we **enumerated all size-k subgraphs** in the input graph
- **Next step: Count the graphs**

Classify subgraphs placed in the ESU-Tree leaves **into non-isomorphic size-k classes:**

  - Determine which subgraphs in ESU-Tree leaves are **topologically equivalent** (**isomorphic**) and group them into subgraph classes accordingly
  - Use McKay's nauty algorithm [McKay 1981]

Graph Isomorphism

- Graphs G and H are **isomorphic** if there exists a bijection $f: V(G) \to V(H)$ such that:
  - Any two nodes **u and v of G are adjacent in G** iff **$f(u)$ and $f(v)$ are adjacent in H**
- Example: Are G and H **topologically equivalent**?



Need to check 9! possible bijections between node sets

Hard computational problem!

$f:$

A	4
B	7
C	1
D	3
E	5
F	8
G	2
H	9
I	6

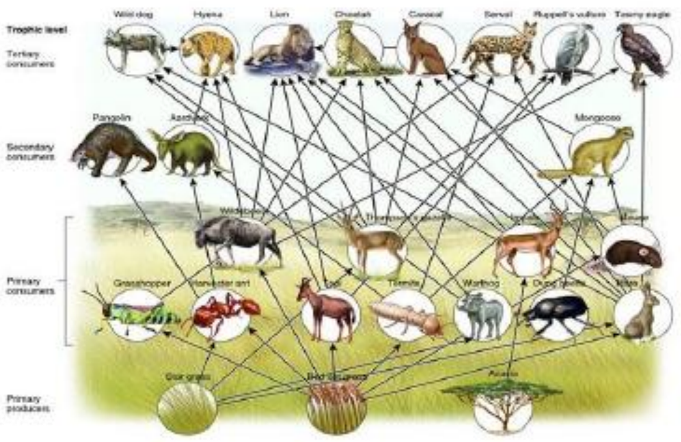
G and H are isomorphic!

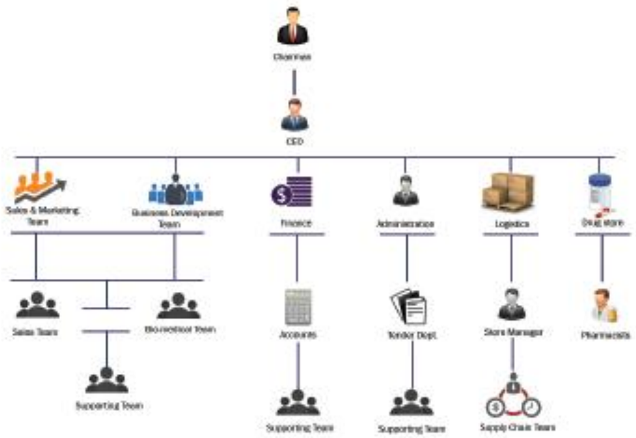
04

Structure Roles vs Community

What are Roles?

- **Roles are "functions" of nodes in a network:**
  - Roles of **species** in **ecosystems**
  - Roles of **individuals** in **companies**





- **Roles are measured by structural behaviors:**
  - centers of stars
  - members of cliques
  - peripheral nodes, etc.

Roles vs Groups

- **Role:** A collection of nodes which have similar positions in a network:
  - Roles are based on the similarity of ties between subsets of nodes
  - Different from **groups/communities**
    - Group is formed based on adjacency, proximity or reachability
    - This is typically adopted in current data mining

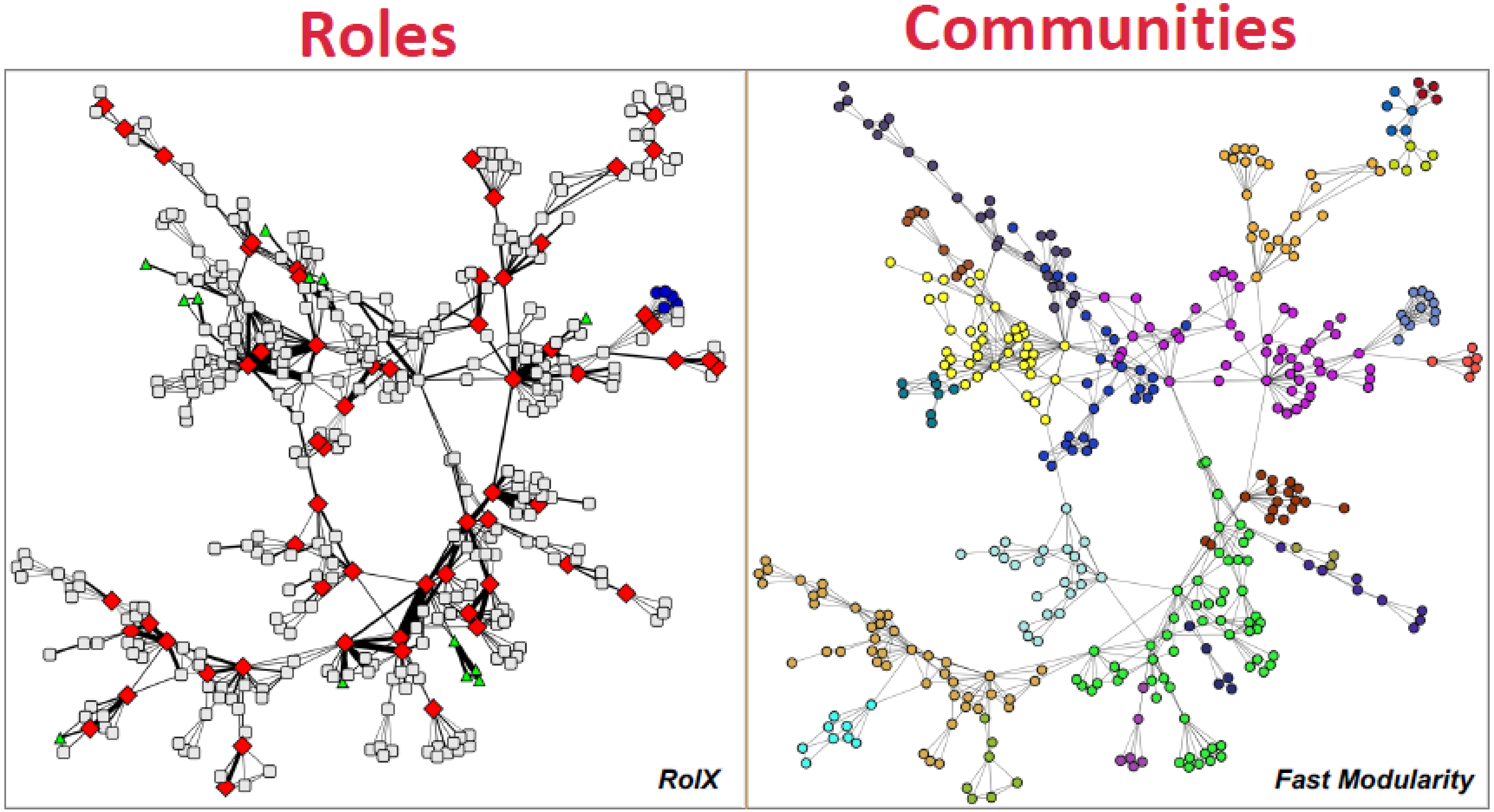
Nodes with the same role need not be in direct, or even indirect interaction with each other

Roles vs Groups

- **Roles:**
  - A group of nodes with similar structural properties
- **Communities/Groups:**
  - A group of nodes that are well-connected to each other
- Roles and communities **are complementary**

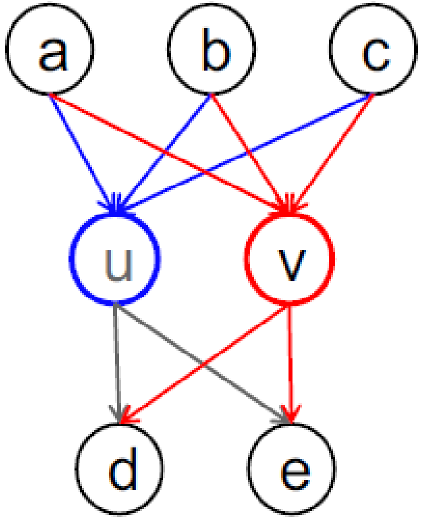
- Consider the social network of a CS Dept:
  - **Roles:** Faculty, Staff, Students
  - **Communities:** AI Lab, Info Lab, Theory Lab

Roles vs Groups



Roles: More Formally

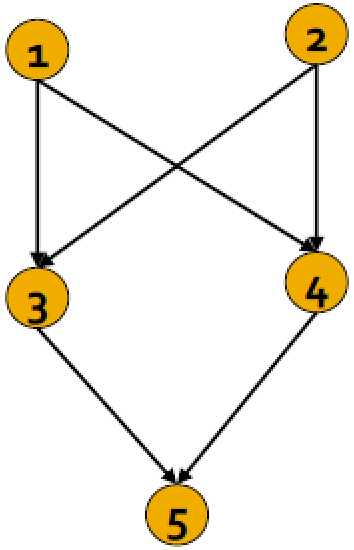
- **Structural equivalence:** Nodes u and v are structurally equivalent if they have the same relationships **to all other nodes** [Lorrain & White 1971]
  - Structurally equivalent nodes are likely to be similar in other ways – *i.e.*, friendships in social networks



Structural Equivalence Example

- Nodes u and v are **structurally equivalent:**
  - For all the other nodes k, node u has tie to k iff node v has tie to k

- **Example:**

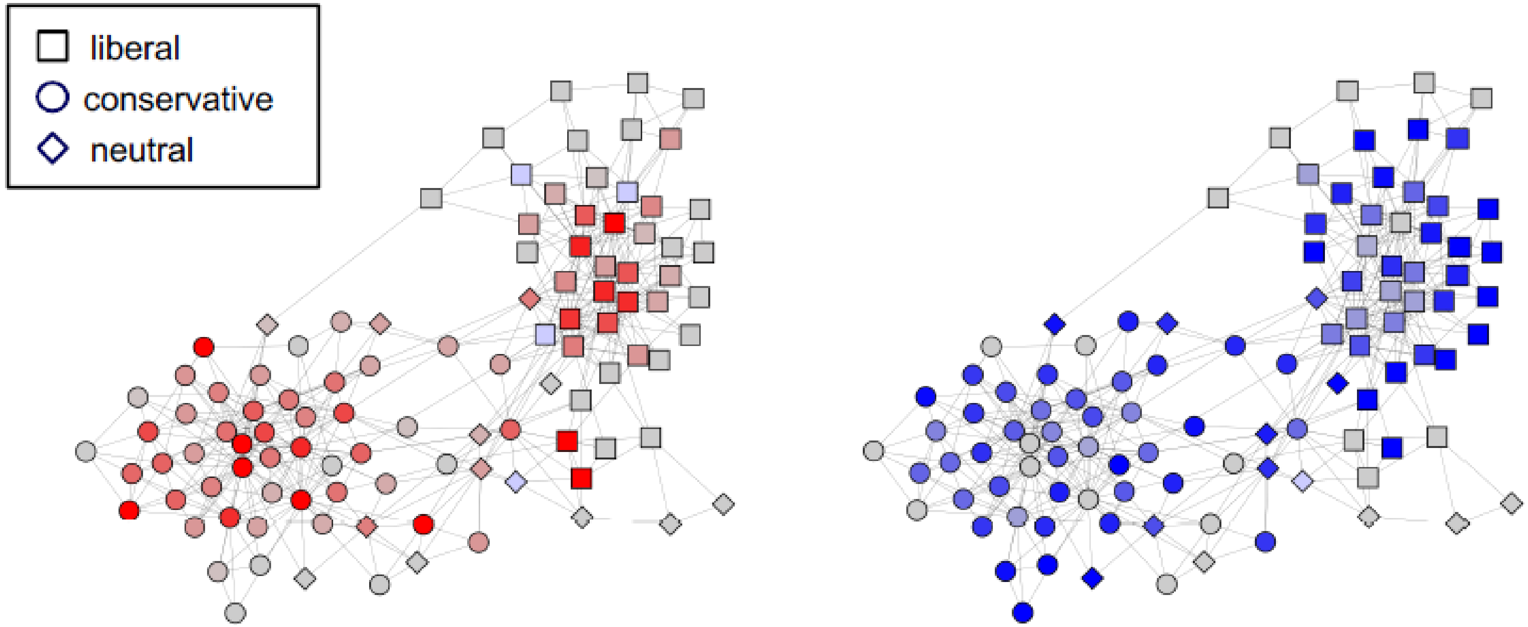


Adjacency matrix

	1	2	3	4	5
1	-	0	1	1	0
2	0	-	1	1	0
3	0	0	-	0	1
4	0	0	0	-	1
5	0	0	0	0	-

- *E.g.*, nodes 3 and 4 are structurally equivalent

Example



Bright **red** nodes are locally **central** nodes

Bright **blue** nodes are **peripheral** nodes

Book labels (i.e., liberal, conservative, neutral) were not given to role discovery algorithm

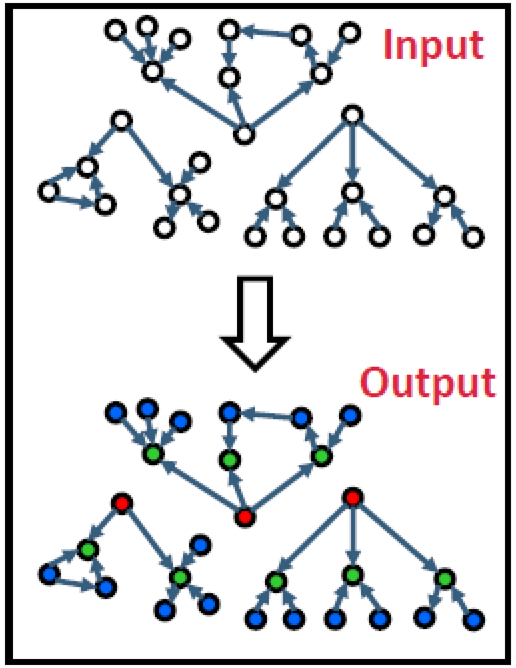
Why Are Roles Importantn?

Task	Example Application
Role query	Identify individuals with similar behavior to a known target
Role outliers	Identify individuals with unusual behavior
Role dynamics	Identify unusual changes in behavior
Identity resolution	Identify, de-anonymize, individuals in a new network
Role transfer	Use knowledge of one network to make predictions in another another
Network comparison	Compute similarity of networks, determine compatibility for knowledge transfer

Structural Role Discovery Method

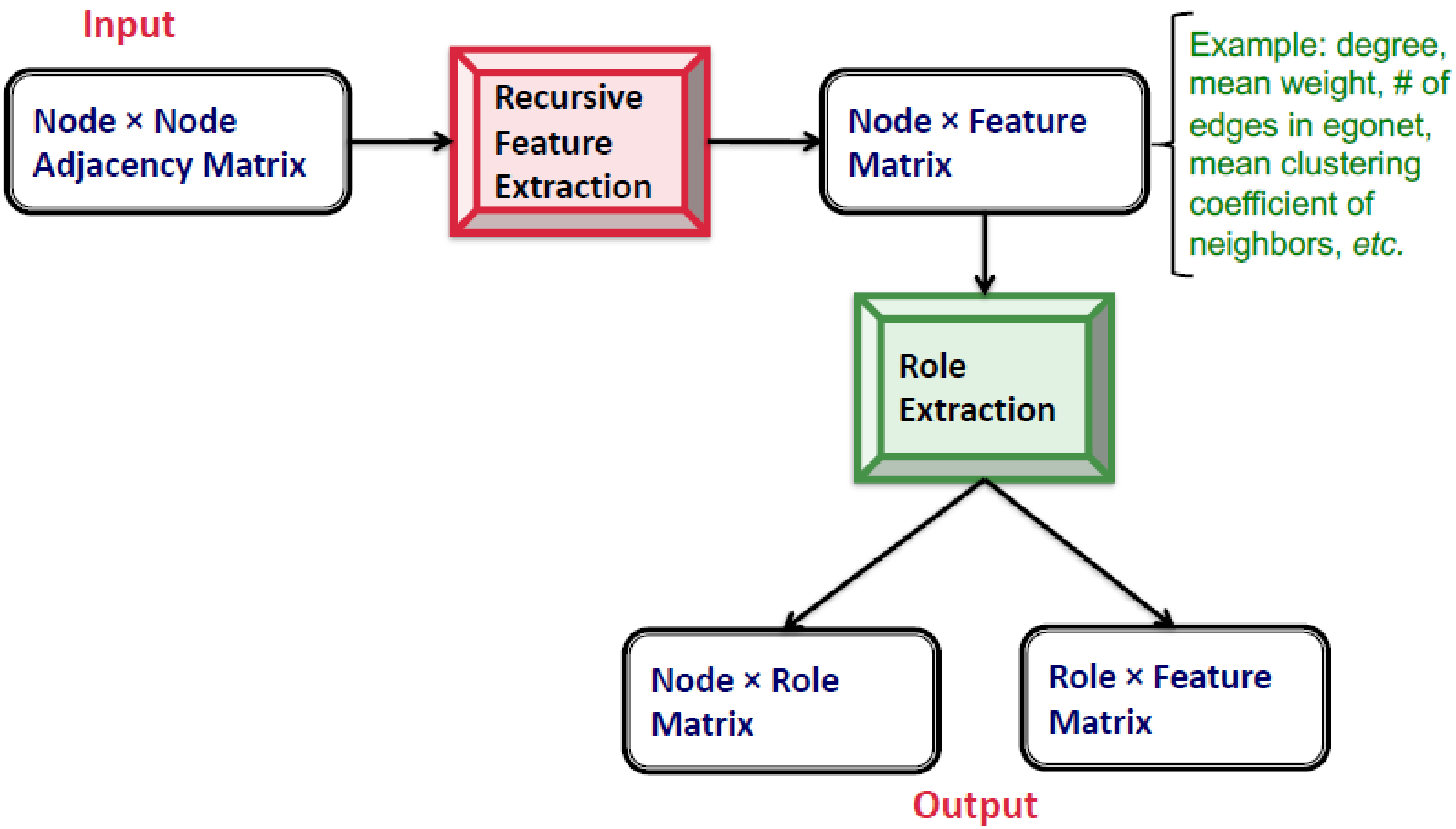
- **RolX:** Automatic discovery of nodes' structural roles in networks [Henderson, et al. 2011b]
  - Unsupervised learning approach
  - No prior knowledge required
  - Assigns a mixed-membership of roles to each node
  - Scales linearly in #(edges)

Role Discovery



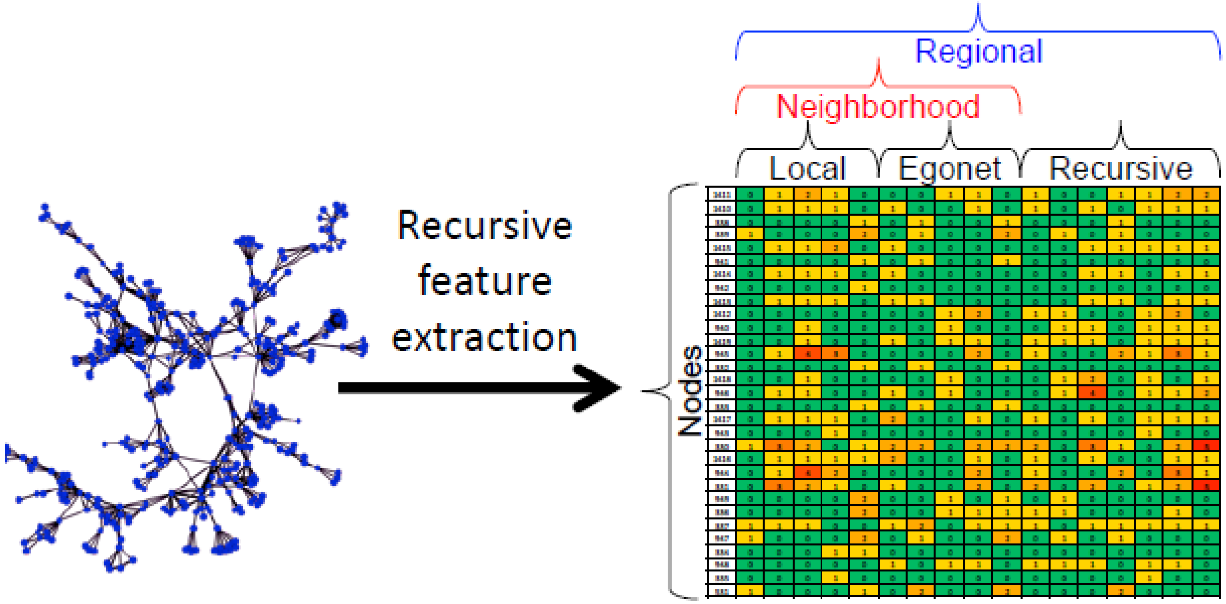
✓Automated discovery
✓Behavioral roles
✓Roles generalize

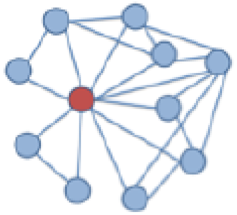
RolX: Approach Overview



Recursive Feature Extraction

- **Recursive feature extraction** [Henderson, et al. 2011a] turns network connectivity into structural features





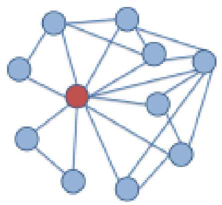
Egonet for **red node**

- **Neighborhood features:** What is a node's connectivity pattern?
- **Recursive features:** To what **kinds** of nodes is a node connected?

Recursive Feature Extraction

- **Idea:** Aggregate features of a node and use them to **generate new recursive features**

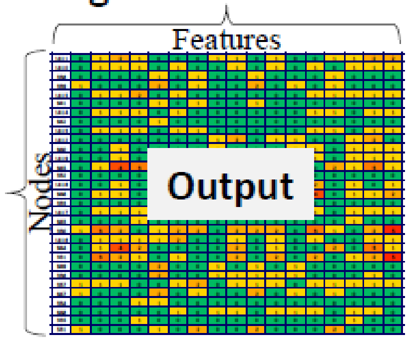
- **Base set of a node's neighborhood features:**
  - **Local features:** All measures of the node degree:
    - If network is directed, include in- and out-degree, total degree
    - If network is weighted, include weighted feature versions
  - **Egonet features:** Computed on the node's egonet:
    - **Egonet** includes the node, its neighbors, and any edges in the induced subgraph on these nodes
    - #(within-egonet edges), #(edges entering/leaving egonet)



Egonet for red node

Recursive Feature Extraction

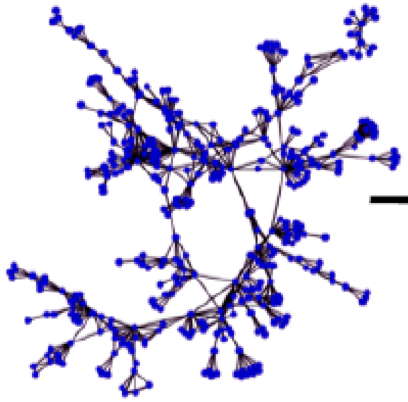
- Start with the base set of node features
- Use the **set of current node features** to generate **additional features**:
  - Two types of **aggregate functions:** mean and sum
    - *E.g.*, mean value of “unweighted degree” feature between all neighbors of a node
    - Compute means and sums over all current features, including other recursive features
  - Repeat



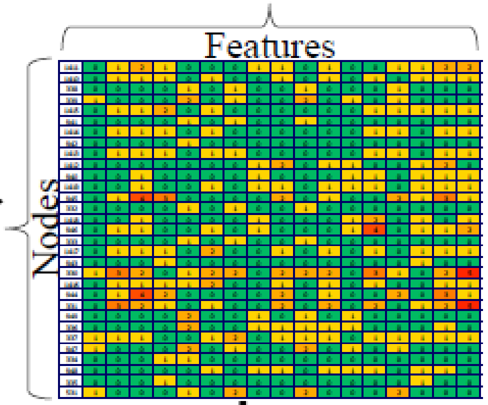
- The number of possible recursive features **grows exponentially** with each recursive iteration:
  - Reduce the number of features using a **pruning technique**:
    - Look for pairs of features that are highly correlated
    - Eliminate one of the features whenever two features are correlated above a user-defined threshold

Role Extraction

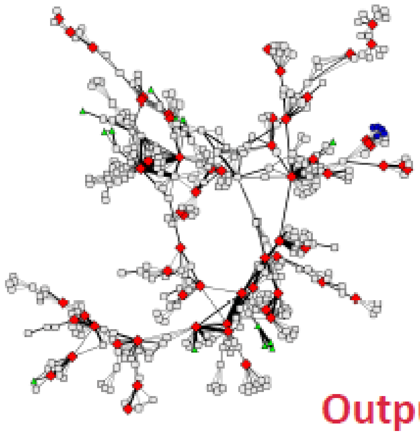
Input



Recursively extract features



Cluster nodes based on extracted features



Output

RolX uses non negative matrix factorization for clustering, MDL for model selection, and KL divergence to measure likelihood

Any Question?